Evolution of a Tri-State Frequency Discriminator for the Source Identification Problem using Genetic Programming

John R. Koza
Computer Science Dept.
258 Gates Building
Stanford University
Stanford, California 94305-9020
koza@cs.stanford.edu
http://www-cs-faculty.stanford.edu/~koza/

Forrest H Bennett III
Visiting Scholar
Computer Science Dept.
Stanford University
Stanford, California 94305
fhb3@slip.net

Jason Lohn
Visiting Scholar
Computer Science Dept.
Stanford University
Stanford, California 94305
jlohn7@leland.stanford.edu

Frank Dunlap
Dunlap Consulting
Palo Alto, California

Martin A. Keane
Martin Keane Inc.
5733 West Grover
Chicago, Illinois 60630
makeane@ix.netcom.com

David Andre
Computer Science Dept.
University of California
Berkeley, California
dandre@cs.berkeley.edu

KEYWORDS

Genetic programming, Automated design of analog electrical circuits, Source identification problem, Frequency discrimination

ABSTRACT

Automated synthesis of analog electronic circuits is recognized as a difficult problem. Genetic programming was used to evolve *both* the topology and the sizing (numerical values) for each component of a circuit that can perform source identification by correctly classify an incoming signal into categories.

1. Introduction

The problem of source identification involves correctly classifying an incoming signal into a category that identifies the source of the signal. The problem is difficult because it requires discovery of the features that distinguish the sources and because no two real-world signals from the same source are ever exactly the same.

This paper considers the problem of evolving the design for an analog electrical circuit that can solve the problem of source identification for incoming signals coming from sources that each emit signals lying in a certain limited range of frequencies. Each incoming signal is to be identified as coming from the first source, the second source, or neither source.

Solving this source identification problem will involve evolving the design of an analog electrical circuit that satisfies specified goals. Considerable progress has been made in automating the design of certain categories of purely digital circuits; however, the design of analog circuits and mixed analog-digital circuits has not proved to be as amenable to automation. As Aaserud and Nielsen (1995) observe,

> "Analog designers are few and far between. In contrast to digital design, most of the analog circuits are still handcrafted by the experts or so-called 'zahs' of analog design. The design process is characterized by a combination of experience and intuition and requires a thorough knowledge of the process characteristics and the detailed specifications of the actual product.
>
> "Analog circuit design is known to be a knowledge-intensive, multiphase, iterative task, which usually stretches over a significant period of time and is performed by designers with a large portfolio of skills. It is therefore considered by many to be a form of art rather than a science."

This paper shows that a design for an analog electrical circuit for a tri-state source identifier (including both the circuit topology and component sizing) can be evolved using genetic programming.

2. Evolution of Circuits

Genetic programming is an extension of John Holland's genetic algorithm (1975) in which the population consists of computer programs of varying sizes and shapes (Koza 1992, 1994a, 1994b; Koza and Rice 1992). Recent research on genetic programming is described in Kinnear (1994), Angeline and Kinnear (1996), and Koza, Goldberg, Fogel, and Riolo (1996).

Genetic algorithms have been applied to the problem of circuit synthesis in the past. A CMOS operational amplifier (op amp) circuit was designed using a modified version of the genetic algorithm (Kruiskamp and Leenaerts 1995); however, the topology of each op amp was one of 24 pre-selected topologies based on the conventional human-designed stages of an op amp. Thompson (1996) used a genetic algorithm to evolve a frequency discriminator on a Xilinx 6216 reconfigurable digital gate array operating in analog mode.

Genetic programming evolves computer programs that are represented as rooted, point-labeled trees with ordered branches. If a mapping can be established between the program trees found in genetic programming and the line-labeled cyclic graphs germane to circuits, genetic programming can then be applied to the problem of evolving the design of a circuit. Cellular encoding (Gruau 1996) enables genetic programming to evolve a neural network.

The principles of developmental biology suggest a way to map program trees into circuits. The starting point of the growth process can be a very simple embryonic electrical circuit. This embryo contains certain fixed and invariant elements for the circuit that is to be designed (e.g., the number of inputs and outputs) as well as certain wires that are capable of subsequent modification. An electrical circuit is progressively developed by applying the functions in a circuit-constructing program tree to the modifiable wires of the embryonic circuit (and to the modifiable wires and components of successor circuits).

The functions in the circuit-constructing program trees include (1) connection-modifying functions that modify the topology of the circuit, (2) component-creating functions that insert components into the circuit, (3) arithmetic-performing functions that appear in arithmetic-performing subtrees as argument(s) to the component-creating functions and that specify the numerical value of the component, and possibly (4) calls to automatically defined functions (if used).

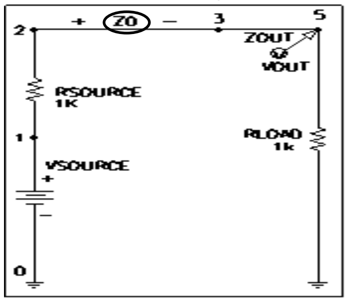

Figure 1 Embryonic circuit.

The developmental process for converting a program tree into a circuit begins with an embryonic circuit. Figure 1 shows a one-input, one-output embryonic circuit. This embryo contains a voltage source VSOURCE connected to node 0 (ground) and 1, a fixed source resistor RSOURCE between nodes 1 and 2, a modifiable wire Z0 between nodes 2 and 3, a fixed isolating wire ZOUT between nodes 3 and 5, a fixed output point (voltage probe) VOUT at node 5, and a fixed load resistor RLOAD between nodes 5 and ground. Only the modifiable wire Z0 is subject to modification during the developmental process.

Each circuit-constructing program tree in the population contains component-creating functions and connection-modifying functions. Each connection-modifying function in a program tree points to an associated highlighted component and modifies the topology of the developing circuit. Each branch of the program tree is created in accordance with a constrained syntactic structure. Branches are composed from construction-continuing subtrees that continue the developmental process and arithmetic-performing subtrees that determine the numerical value of components. Connection-modifying functions have one or more construction-continuing subtrees, but no arithmetic-performing subtrees. Component-creating functions have one construction-continuing subtree and typically have one arithmetic-performing subtree. This constrained syntactic structure is preserved by using structure-preserving crossover with point typing.

Component-creating functions insert a component into the developing circuit and assigns component value(s) to the component. Each component-creating function has a writing head that points to an associated highlighted component in the developing circuit and modifies the highlighted component in a specified way. The construction-continuing subtree of each component-creating function points to a successor function or terminal in the circuit-constructing program tree.

The arithmetic-performing subtree of a component-creating function consists of a composition of arithmetic functions (addition and subtraction) and random constants (in the range –1.000 to +1.000) and specify the numerical value of a component.

Space does not permit a detailed description of each function. See Koza, Andre, Bennett, and Keane (1996), and Koza, Bennett, Andre, and Keane (1996a, b, c, d).

3. Preparatory Steps

The goal is to evolve the design for an RLC circuit that classifies the incoming signal into three categories. Specifically, the desired circuit is to produce an output of 1/2 volt (plus or minus 240 millivolts) if the frequency of the incoming signal is within 10% of 256 Hz, produce an output of 1 volt (plus or minus 240 millivolts) if the frequency of the incoming signal is within 10% of 2,560 Hz, and otherwise produce an output of 0 volts (plus or minus 240 millivolts). The tolerance of 240 (rather than 250) millivolts was chosen to avoid the possibility of a tie.

Before applying genetic programming to a problem of circuit synthesis, the user must perform seven major preparatory steps, namely (1) identifying the embryonic circuit, (2) determining the architecture of the overall circuit-constructing program trees, (3) identifying the terminals of the programs, (4) identifying the primitive functions contained in the programs, (5) creating the fitness measure, (6) choosing certain control parameters (notably population size and the maximum number of generations to be run), and (7) determining the termination criterion and method of result designation.

The one-input, one-output embryonic circuit of figure 1 was used for this problem.

Since the embryonic circuit has one modifiable wire, there is one writing head and one result-producing branch in each circuit-constructing program tree.

For this problem, the function set, F_{CCS}, for each construction-continuing subtree is

F_{CCS} = {`R`, `L`, `C`, `SERIES`, `PSS`, `PSL`, `FLIP`, `NOP`, `SAFE_CUT`, `T_PAIR_CONNECT_0`, `T_PAIR_CONNECT_1`}.

The terminal set, T_{CCS}, for each construction-continuing subtree consists of

T_{CCS} = {`END`}.

The function set, Faps, for each arithmetic-performing subtree is

Faps = {`+`, `-`}.

The terminal set, Taps, for each arithmetic-performing subtree consists of

Taps = {$\Re$}.

$\Re$ represents random constants from –1.0 to +1.0.

The evaluation of fitness for each individual circuit-constructing program tree in the population begins with its execution. This execution applies the functions in the program tree to the embryonic circuit, thereby developing the embryonic circuit into a fully developed circuit. A netlist describing the fully developed circuit is then created. The netlist identifies each component of the circuit, the nodes to which that component is connected, and the value of that component. Each circuit is then simulated to determine its behavior. The 217,000-line SPICE (Simulation Program with Integrated Circuit Emphasis) simulation program (Quarles et al. 1994) was modified to run as a submodule within the genetic programming system. For this problem, the voltage `VOUT` is probed at node `5` and the circuit is simulated in the frequency domain. SPICE is requested to perform an AC small signal analysis and to report the circuit's behavior for each of 101 frequency values chosen over four decades of frequency (between 1 and 10,000 Hz). Each decade is divided into 25 parts (using a logarithmic scale).

Fitness is measured in terms of the sum, over these 101 fitness cases, of the absolute weighted deviation between the actual value of the output voltage at the probe point `VOUT` and the target value for voltage. The smaller the value of fitness, the better.

The three points that are closest to the band located within 10% of 256 Hz are 229.1 Hz, 251.2 Hz, and 275.4 Hz. The procedure for each of these three points is as follows: If the voltage equals the ideal value of 1/2 volts in this interval, the deviation is 0.0. If the voltage is within 240 millivolts of 1/2 volts, the absolute value of the deviation from 1/2 volts is weighted by a factor of 20. If the voltage is more than 240 millivolts from 1/2 volts, the absolute value of the deviation from 1/2 volts is weighted by a factor of 200. This arrangement reflects the fact that the ideal output voltage for this range of frequencies is 1/2 volts, the fact that a 240 millivolts discrepancy is acceptable, and the fact that a larger discrepancy is not acceptable.

The three points that are closest to the band located within 10% of 2,560 Hz are 2,291 Hz, 2,512 Hz, and 2,754 Hz. The procedure for each of these three points is as follows: If the voltage equals the ideal value of 1 volt in this interval, the deviation is 0.0. If the voltage is within 240 millivolts of 1 volt, the absolute value of the deviation from 1 volt is weighted by a factor of 20. If the voltage is more than 240 millivolts from 1 volt, the absolute value of the deviation from 1 volt is weighted by a factor of 200.

The procedure for each of the remaining 95 points is as follows: If the voltage equals the ideal value of 0 volts, the deviation is 0.0. If the voltage is within 240 millivolts of 0 volts, the absolute value of the deviation from 0 volts is weighted by a factor of 1.0. If the voltage is more than 240 millivolts from 0 volts, the absolute value of the deviation from 0 volt is weighted by a factor of 10.

Circuits that cannot be simulated by SPICE are assigned a high penalty value of fitness (10^8).

Hits are defined as the number of fitness cases (0 to 101) for which the voltage is acceptable or ideal.

The population size was 640,000. The percentage of genetic operations on each generation was 89% one-offspring crossovers, 10% reproductions, and 1% mutations. The maximum size for the result-producing branch was 600 points. Other parameters were the default values specified in Koza 1994 (appendix D).

This problem was run on a medium-grained parallel Parsytec computer system consisting of 64 80 MHz Power PC 601 processors arranged in a toroidal mesh with a host PC Pentium type computer. The distributed genetic algorithm was used with a population size of Q = 10,000 at each of the D = 64 demes. On each generation, four boatloads of emigrants, each consisting of B = 2% (the migration rate) of the node's subpopulation (selected on the basis of fitness) were dispatched to each of the four toroidally adjacent processing nodes (Andre and Koza 1996).

4. Results

The best circuit from generation 0 (figure 2) has a fitness of 286.2 and 64 hits. It has no inductors, two capacitors, and two resistors (in addition to the source and load resistors in the embryo). Figure 5 shows the behavior of the best circuit of generation 0 in the frequency domain. The horizontal axis is logarithmic and ranges between 1 and 10,000 Hz. The vertical axis ranges between zero and 1 volt. Notice that this circuit pays no special attention to the frequencies around 256 and 2,560 Hz.

The best circuit from generation 20 (figure 3) has a fitness of 129.1 and 76 hits. Figure 6 shows its behavior. Notice the emergence of two distinct areas around 256 and 2,560 Hz.

The best circuit from generation 106 (figure 4) achieves a fitness of 21.4 and 101 hits. It has seven inductors, 15 capacitors, and four resistors. Its circuit-constructing program tree has 551 points. Figure 7 shows its behavior in the frequency domain. As can be seen, the circuit will produce an output voltage in the correct band for incoming signals emanating from the first source, the second source, or neither.

The run took 43 hours and processed about 67,840,000 individuals. The 64 80 mega Hertz processors operate together at a rate of 5.12 giga Hertz, so that there were about 8×10^{14} clock cycles in the run. If one clock cycle is approximately equal to a computer operation, this is about 10^{15} operations. Noting that the human brain has about 10^{12} neurons operating at an approximately millisecond rate, we have designated 10^{15} operations as a *brain second*.

Interestingly, approximately 1 *bs* of computational effort was also required to evolve a one-dimensional cellular automata rule for the majority classification task whose accuracy (82.326%) exceeds that of the original 1978 Gacs-Kurdyumov-Levin (GKL) rule, all other known subsequent human-written rules, and all other known rules produced by automated approaches for this problem (Andre, Bennett, and Koza 1996).

Moreover, the performance of four different versions of genetic programming (Koza 1994a, Koza and Andre 1996a) on the transmembrane segment identification problem is slightly superior to that of algorithms written by knowledgeable human investigators. Again, approximately 1 *bs* of computational effort was required to produce each of these four results.

In addition, approximately 1 *bs* of computational effort was required for the runs of genetic programming that successfully evolved protein motifs for detecting the D-E-A-D box family of proteins and for detecting the manganese superoxide dismutase family as well or better than the comparable human-written motifs found in the PROSITE database (Koza and Andre 1996c).

5. Conclusion

Automated synthesis of analog electronic circuits is recognized as a difficult problem. We have shown that genetic programming successfully evolved the design for a circuit that can perform source identification by correctly classify an incoming signal into categories.

References

Aaserud, O. and Nielsen, I. Ring. 1995. Trends in current analog design: A panel debate. *Analog Integrated Circuits and Signal Processing.* 7(1) 5-9.

Andre, David, Bennett III, Forrest H, and Koza, John R. 1996. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In Koza, John R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L. (editors). *Genetic Programming 1996: Proceedings of the First Annual Conference.* Cambridge, MA: The MIT Press.

Andre, David and Koza, John R. 1996. Parallel genetic programming: A scalable implementation using the transputer architecture. In Angeline, P. J. and Kinnear, K. E. Jr. (editors). 1996. *Advances in Genetic Programming 2.* Cambridge: MIT Press.

Angeline, Peter J. and Kinnear, Kenneth E. Jr. (editors). 1996. *Advances in Genetic Programming 2.* Cambridge, MA: The MIT Press.

Gruau, Frederic. 1996. Artificial cellular development in optimization and compilation. In Sanchez, Eduardo and Tomassini, Marco (editors). 1996. *Towards Evolvable Hardware.* Lecture Notes in Computer Science, Vol. 1062. Berlin: Springer-Verlag. 48–75.

Holland, John H. 1975. *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press.

Kinnear, Kenneth E. Jr. (editor). 1994. *Advances in Genetic Programming.* Cambridge, MA: MIT Press.

Koza, John R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: MIT Press.

Koza, John R. 1994a. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: MIT Press.

Koza, John R. 1994b. *Genetic Programming II Videotape: The Next Generation.* : MIT Press.

Koza, John R. and Andre, David. 1996a. Classifying protein segments as transmembrane domains using architecture-altering operations in genetic programming. In Angeline, Peter J. and Kinnear, Kenneth E. Jr. (editors). 1996. *Advances in Genetic Programming II.* Cambridge, MA: MIT Press.

Koza, John R. and Andre, David. 1996b. Evolution of iteration in genetic programming. In *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming.* Cambridge, MA: MIT Press.

Koza, John R. and Andre, David. 1996c. Automatic discovery of protein motifs using genetic programming. In Yao, Xin (editor). 1996. *Evolutionary Computation: Theory and Applications.* Singapore: World Scientific. In Press.

Koza, John R., Andre, David, Bennett III, Forrest H, and Keane, Martin A. 1996. Use of automatically defined functions and architecture-altering operations in automated circuit synthesis using genetic programming. In Koza, John R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference.* Cambridge, MA: MIT Press.

Koza, John R., Bennett III, Forrest H, Andre, D., and Keane, M. A. 1996a. Toward evolution of electronic animals using genetic programming. *Artificial Life V: Proceedings* Cambridge, MA: The MIT Press.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1996b. Four problems for which a computer program evolved by genetic programming is competitive with human performance. *Proceedings of the 1996 IEEE*

International Conference on Evolutionary Computation. IEEE Press. Pages 1–10.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1996c. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In Gero, John S. and Sudweeks, Fay (editors). *Artificial Intelligence in Design '96*. Dordrecht: Kluwer. Pages 151-170.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1996d. Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming. In Koza, John R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference.* Cambridge, MA: MIT Press.

Koza, John R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference.* Cambridge, MA: The MIT Press.

Koza, J R., and Rice, J. 1992.*Genetic Programming: The Movie*. Cambridge, MA: MIT Press.

Kruiskamp Marinum W. and Leenaerts, Domine. 1995. DARWIN: CMOS opamp synthesis by means of a genetic algorithm. *Proceedings of the 32nd Design Automation Conference*. New York, NY: Association for Computing Machinery. 433–438.

Quarles, Thomas, Newton, A. R., Pederson, D. O., and Sangiovanni-Vincentelli, A. 1994. *SPICE 3 Version 3F5 User's Manual*. Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA. March 1994.

Thompson, Adrian. 1996. Silicon evolution. In Koza, John R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference.* Cambridge, MA: MIT Press.

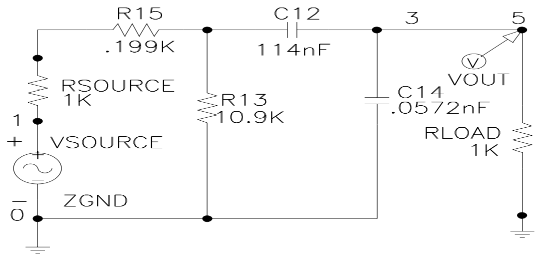

Figure 2 Best circuit of generation 0.

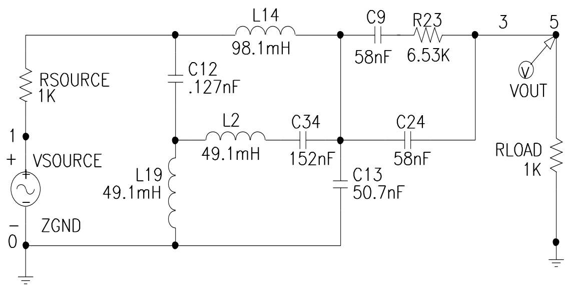

Figure 3 Best circuit of generation 20.

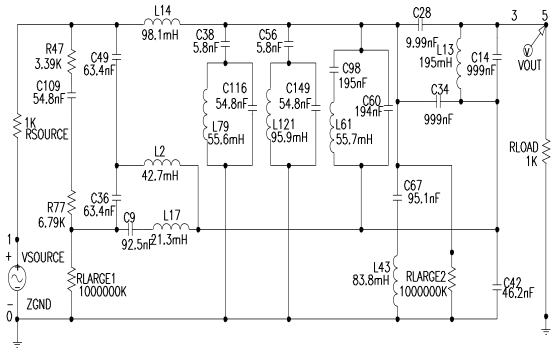

Figure 4 Best circuit of generation 106.

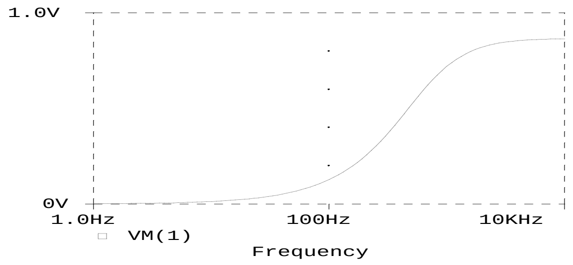

Figure 5 Frequency domain behavior of the best circuit of generation 0.

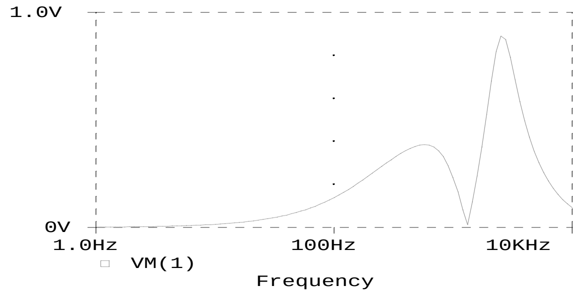

Figure 6 Frequency domain behavior of the best circuit of generation 20.

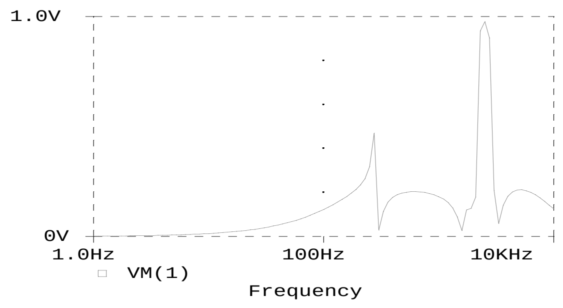

Figure 7 Frequency domain behavior of the best circuit of generation 106.

Camera-Ready Version – Submitted January 6, 1997 to First International Workshop on Frontiers in Evolutionary Algorithms (FEA-97) to be held on February 28 – March 2 – 5, 1997 in Research Triangle Park, North Carolina.

Evolution of a Tri-State Frequency Discriminator for the Source Identification Problem using Genetic Programming

John R. Koza
Computer Science Dept.
258 Gates Building
Stanford University
Stanford, California 94305-9020
koza@cs.stanford.edu
http://www-cs-faculty.stanford.edu/~koza/

Forrest H Bennett III
Visiting Scholar
Computer Science Dept.
Stanford University
Stanford, California 94305
fhb3@slip.net

Jason Lohn
Visiting Scholar
Computer Science Dept.
Stanford University
Stanford, California 94305
jlohn7@leland.stanford.edu

Frank Dunlap
Dunlap Consulting
Palo Alto, California

Martin A. Keane
Martin Keane Inc.
5733 West Grover
Chicago, Illinois 60630
makeane@ix.netcom.com

David Andre
Computer Science Dept.
University of California
Berkeley, California
dandre@cs.berkeley.edu